Thesis Proposal:

OnStar for Bikes: Automatic collision detection and response using embedded devices

Justin A. S. Bull
justin.bull@ryerson.ca
Department of Computer Science,
Ryerson University

September 28, 2013

1 A Cyclist's Problem

One of the crucial moments for a cyclist is immediately after a collision, unfortunately it's also one of the most confusing. When a cyclist is involved in any sort of collision, they're subject severe injury, psychological shock, or being taken advantage of. More often than not, what they need is medical attention and police involvement. There is currently a problem with technology being under-utilized to provide a service to cyclists when they're dazed and confused seconds after a collision.

What the average cyclist needs is help to already be on the way while they orient themselves and try move to safe spot to rest, if they can. I believe this is a significant problem for any urban commuter that uses cycling as a mode of transportation. Furthermore, I believe this is a problem that no one has really attempted to solve using technology as a tool.

Having been in a few collisions myself, I can speak of their powerfully disorientating nature and I'm personally motivated by the solution presented in this proposal.

2 A Tangible Solution

To this problem I propose a potential solution: *OnStar for Bikes*. By using technological advances in the past decade, *OnStar for Bikes*, at its core, would use the Global Positioning System (GPS)[1], 3-dimensional accelerometers, and the Short Message Service (SMS)[2] to detect a collision and immediately communicate with an emergency contact, providing the cyclist's location and other relevant data so that they may receive the help they so desperately need. It'd be a simple set-and-forget device, accessible to the average commuting cyclist for under \$100 and an additional trivial monthly service fee.[3]

The *OnStar for Bikes* project would be comprised of two components: an embedded device (the device), and a web-application interface (the web app).

2.1 The Device

The aim is to make a physical device that's small, resistant to theft, with a long battery life, and easily mountable onto a bike frame. It'd use the SMS to dispatch Application Programming Interface (API)[4] calls to the web-app component who's responsible for contacting a phone number (via text-to-speech or text message) or e-mail address. Upon detecting a collision, the device will also begin to emit a continuous beep, informing the cyclist that it's been activated and their previously configured emergency contact has been contacted.

[1] `http://en.wikipedia.org/wiki/Global_Positioning_System`

[2] `http://en.wikipedia.org/wiki/Short_Message_Service`

[3] Due to the implementation strategy a monthly fee would have to be charged in order to recoup the costs of an activate phone line.

[4] `http://en.wikipedia.org/wiki/Application_programming_interface`

2.1.1 API Communication

Twilio, a cloud communications company, provides a service where you can programmatically send and receive SMS messages or phone calls using a 3rd party API.[5] *Twilio*'s service would be the backbone for communication between the device and the web app.

Every API request to the web app would fit under 140 characters (140 bytes), as would their responses. Presently, the core calls would be:

1. `collision`: A collision has occurred at a GPS coordinate. The device provide its collision type guess.
2. `low-bat`: The battery on the device is low and requires recharging.
3. `beep`: The device is instructed to emit a beeping sound from the web app.

The device would have bi-directional communication, capable of sending calls based on sensory data (e.g. `collision`) or performing actions based on SMS messages from the web app (e.g. `beep`).

2.1.2 Collision Detection

Using collision data collected from the Toronto Police Service, I'd research what G-forces typically define a 'collision'. Additionally, personal experimentation would be performed to see what normal G-forces are generated from every day biking. Based on these two data sources, I should be able to define a threshold the device uses to determine whether or not a collision has occurred.

I plan to have the device guess at what type of collision the cyclist was involved in, be it a dangerous fall, "T-bone", "head-on", or "rear-end" collision with another vehicle. This is where the benefit of a 3-dimensional accelerometer would come into play (offering magnitude and direction), as opposed to a simple accelerometer that provides just magnitude.

2.2 The Web App

Written in Ruby[6] (using the Ruby on Rails framework[7]) the web application would be responsible for an API back-end for the device and graphical front-end for the user.

2.2.1 The Back-End API

An API would be exposed such that the device is capable of communicating to the web app and vice versa using *Twilio* as middleware. Essentially, the back-end API's sole purpose is to allow the device to control the web app.

[5] `http://www.twilio.com/sms/api`
[6] `https://www.ruby-lang.org/en/`
[7] `http://rubyonrails.org/`

Since the embedded device is limited in capability, it's API may not support the verbosity or variable-length nature of popular data-interchange formats such as JSON[8] or XML.[9] As a result, I will define a custom API message structure (the payload) that *Twilio* would deliver to the device to perform an action. It'd be fixed-length and its detailed design would make itself apparent when the prototype is constructed.

2.2.2 The Front-End GUI

A simple interface available on the World Wide Web for a user to login to and register their device, configure emergency contacts & corresponding message to be sent, and view previously detected collisions plotted on a map using the collected GPS data.

An additional feature would be a "Find my Bike" whereby the user can click a button on the web app and the device will emit a beep for 45 seconds.

3 Prototype and Technical Document

3.1 Prototype

For the thesis, I would build a device prototype and minimum viable product web application. The device would be able to perform the basic function of detecting a collision and reporting it to the web app and the web app would be capable of informing an emergency contact. Current plans is to use an Arduino prototyping platform[10] for the device.

3.2 Technical Document

The thesis will include a technical document providing a full schematic of the prototype and architecture of the web app. Furthermore, a detailed description of the custom communication API will be defined.

In addition to the specifics and implementation details of the prototype, the G-force data collected from the TPS and personal experiments will be presented. Conclusions and design decisions from the results will also be described.

4 Required Background Reading & Research

The solution described above is quite a mouthful, utilizing multiple technologies to provide a unique and effective product. The following topics and technologies would have to be researched in order to have enough knowledge to carry out the thesis:

Mention the background reading required for bulk SIM subscription from a carrier like Bell or Rogers

[8] http://www.json.org/
[9] http://en.wikipedia.org/wiki/XML
[10] http://www.arduino.cc/

4.1 Arduino & Embedded C

Arduino uses the C programming language, which I'm already mildly familiar with in the context of personal computers running Linux. However, the Arduino is an embedded device, and I'll have to learn the constraints of writing software in such a limited capacity.

For every component of the device (GPS, GSM, 3D accelerometer, lithium-ion battery, etc.), I'll have to source each component I wish to use, learn about it via its data sheets, and then integrate it into the Arduino board. I presently have no experience or knowledge in embedded GPS chips, full-stack GSM devices, or 3D accelerometers. Background research will go mostly into GSM SMS communications on an embedded device, as that'd be the most challenging and complex.

Furthermore, almost every single component that'd be used in the prototype will require extensive research in how it works, what type is right for this thesis, and how it can be integrated into the device.

4.2 Collision Data

In order to program the device correctly, data on what G-forces are normally exerted on a bike during a collision will be needed. I'll have to establish contact and cooperation of the Toronto Police Service to access their scene reconstruction information in hopes to get an introductory understanding of what occurs during a collision.

4.3 *Twilio* API

The documentation[11] and libraries[12] for the *Twilio* API will have to be read, understood, and applied both in hardware and in software in order for the device to communicate to the web app.

[11] `http://www.twilio.com/docs/api`
[12] `https://github.com/twilio/twilio-ruby`